# Processing Assignment 4: Add the Target

Assignment Due 11/11

## Description

This deliverable will include several new features over the Processing 3 assignment:

1. Draw the target at a random location somewhere down field from the cannon. In the following figure the target is represented as a red line at the bottom of the screen, highlighted with the arrow.
2. Record and present the number of attempts. Notice that “Number of Attempts” is drawn in the upper left hand corner. This is not the same as scoring which will be tackled in the next assignment.
3. Use the left and right arrow keys to adjust the cannon angle.
4. Determine and present whether each shot hits or misses the target. Notice that “Status” is being drawn in the upper left hand corner with the other status text.

## 

## Psudocode

### New Variables

We added the following highlighted variables to the program.

**targetStart** is the randomly generated location of the target on the screen.

**targetWidth** is the width (in pixels) of the target when drawn at the bottom of the screen.

**numAttemts** is the number of times the user fires the cannon.

**status** is a variable that holds the game’s status. It is eventually presents “HIT” or “MISS” depending on whether the target is struck.

// Simulation Constants

float v = 160; // Velocity

float g = 32; // Gravity

float deltaT = .05; // time delta

float cannonAngle = radians(45); // Cannon angle in radians

float firingAngle = radians(45); // firing angle in radians

float maxT, currT; // Simulation Time

**int targetStart;**

**int targetWidth = 25;**

**int numAttempts;**

**String status;**

### Changes to the draw() function

There are two changes to the draw function. The first change is the use of the status variable to determine whether to continue updating the projectile’s position after the simulation ends and the target is checked. Without the check the checkTarget() will continue to be evaluated for each activation of the draw() function. This will become an issue in the final version of the game.

The second change to the draw() function is the addition of the drawTarget() function which is responsible for drawing the target on the screen.

We also will be adding code to the checkTarget() function.

void draw() {

background(255);

**if (status == "run")** {

if (currT > maxT) {

**checkTarget**(); // Stop animation

}

else {

currT = currT + deltaT;

}

}

**drawTarget**();

drawCannon();

drawStatus();

drawShot();

}

### Changes to the setup() function

Remember that Processing calls the setup() function once when the program is run. Setup() is where the application’s initialization is placed. The changes to setup() for this assignment was to randomly generate the location of the target. Notice the use of Processing’s random() function. This function will return a randomly generated number between 100 and the width of the screen. This random number provides the starting location of a line to be drawn on the bottom of the screen that represents the target to be aimed for.

Setup() also sets the number of attempts to zero and sets the status flag to ‘halt’.

**float target**;

void setup() {

size(800, 400);

//frameRate(3);

**targetStart = int(random(100, width-targetWidth));**

**numAttempts** = 0;

**status** = "halt";

}

### Adding the drawTarget() function

The drawTarget() function draws the target on the screen using the randomly generated target location set in the setup() function.

void **drawTarget**() {

stroke(255, 0, 0);

line(targetStart, height-1, **targetStart** + targetWidth, height-1);

stroke(0);

}

### Completing the checkTarget() function

The checkTarget() function calculates the X position (xpos) of the projectile (cannon ball) at the current simulation time (currT). If xpos is within the target range, the shot is scored a HIT, else the shot must be a miss.

void checkTarget() {

float xpos = v \* currT \* cos(firingAngle);

if (**xpos >= targetStart && xpos <= targetStart + targetWidth**) {

status = "HIT";

}

else {

status = "MISS";

}

}

### Changes to fireCannon() function

We change the game status to “RUNNING” when the cannon is fired.

void fireCannon() {

currT = 0;

firingAngle = cannonAngle;

maxT = 2\*v\*sin(firingAngle)/g;

**numAttempts = numAttempts + 1;**

**status = "run";**

}

### Using the Arrow Keys to Adjust the Cannon Angle

This version uses the left and right arrow keys to modify the cannon angle. Detecting the use of the arrow keys is a bit more complex than the use of the angle brackets in the previous version. Instead of comparing the ‘key’ variable to a character, we need to detect that the key is CODED and the keyCode is LEFT as shown in the following if statements. Compare this with the previous example to better understand.

void keyPressed() {

//println("Key Pressed: " + key);

if (**key == CODED && keyCode == LEFT**) {

cannonAngle = cannonAngle + .01;

if (cannonAngle > HALF\_PI) {

cannonAngle = HALF\_PI;

}

}

else if (**key == CODED && keyCode == RIGHT**) {

cannonAngle = cannonAngle - .01;

if (cannonAngle < 0) {

cannonAngle = 0;

}

}

else if (key == ' ') {

fireCannon();

}

}

## Deliverables

Each team will submit their sketchbook directory containing their submission on a USB thumb drive. Just copy the processing directory containing your work onto a USB thumb drive.

Also include a README.txt file on the USB drive containing your section and team numbers.   
**No File..No Grade.**

THE THUMB DRIVE MUST CONTAIN THE FOLDER WHICH IS THE PROCESSING SKETCHBOOK TO BE GRADED AND THE README FILE

Graders will evaluate the submissions by executing the submitted program file and ensuring that it executes correctly.